MCMCBasic

January 21, 2018

In [38]:
```
'''The neccesary packages for our MCMC code'''

import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as st
```

In [39]:
```
'''Calculate the posterior'''
def target(lik, prior, param, theta):
    if theta < 0 or theta > 1:
        return 0
    else:
        return lik(param[0], theta).pmf(param[1])*prior.pdf(theta)
```

In [40]:
```
param = 14,10 #our initial data: 14 tries and 10 heads
a = 1 #Params for the beta function
b = 1

lik = st.binom #Our likelihood
prior = st.beta(a,b) #Our prior
sigma = 0.01 #Standar desviation for the gaussian proposal distribution

naccept = 0. #Accepted steps
theta = 0.1 #Our guest for p

niters = 100000 #How many iterations we want to do
samples = np.zeros(niters+1)
samples[0] = theta

for i in range(niters):
    theta_p = theta + st.norm(0, sigma).rvs()
    rho = min(1, target(lik, prior, param, theta_p)/target(lik, prior, param, theta))
    u = np.random.uniform()
    '''Metropolis Hasting algorithm'''
    if u < rho:
        naccept += 1
        theta = theta_p
    samples[i+1] = theta
nmcmc = len(samples)//2
print ("Efficiency = ", naccept/niters)
```

```
('Efficiency = ', 0.9715)
```

In [47]:
```
'''We can plot our posterior results'''
thetas = np.linspace(0, 1, 200)
post = st.beta(param[1]+a, param[0]-param[1]+b)
ci = post.interval(0.95) #Our confidence interval

plt.style.use('ggplot') #It is only for the style of the plot
plt.figure(figsize=(12, 9))
plt.hist(samples[nmcmc:], 40, histtype='step', normed=True, linewidth=1, label='Distrib
plt.hist(prior.rvs(nmcmc), 40, histtype='step', normed=True, linewidth=1, label='Distri
plt.plot(thetas, post.pdf(thetas), c='red', linestyle='--', alpha=0.5, label='True post
plt.axhline(0.3, ci[0], ci[1], c='black', linewidth=2, label='95% CI');
plt.xlim([0,1]);
plt.legend(loc='best');
plt.show()
```